COSINE MVP System Architecture Document (Updated with L2 Ledger)

Dimoris Chinyui

February 21, 2025

Contents

1. Overview

This updated COSINE MVP system architecture incorporates the new "Decentralized Storage and Updates" concept introduced in the latest COSINE whitepaper. Instead of relying on IPFS as the primary storage layer, the protocol now maintains a custom Layer-2 (L2) ledger managed by validators. This ledger serves as the single source of truth for:

- **Wallet State Trie (WST)**: Storing per-wallet data such as balances, credit scores, and linking status.
- **Global State Object (GSO)**: Holding protocol-wide variables (e.g., penalty/reward scaling factors, total staked amounts).
- **Linked L1 Trie (L1T)**: Enforcing one-to-one mappings between Layer-1 (L1) and Layer-2 (L2) wallets.
- **Block Structure** with headers, transactions, and state updates to ensure consistency and tamper-resistance.

The validator nodes are now responsible for:

- Managing the L2 ledger (including WST, GSO, and L1T).
- Producing blocks containing valid transactions.
- Reaching consensus on the updated state via VRF-based subset selection.

The replacement of IPFS with this custom ledger entails:

- Removing or repurposing the old *Decentralized Data Storage Module.*
- Enhancing the *Validator Node Module* to manage the ledger data structures and block production.
- Updating cross-chain bridging operations in both *Smart Contracts* (for locking/minting on L1) and the new *Bridge Operation Submodule* (for L2 ledger updates) but remember bitcoin is a non-programmable blockchain Cosine token cannot be deployed on the bitcoin blockchain.
- Adjusting *API/CLI Interface* endpoints to allow users to submit transactions and query the ledger state.

All other core functionalities (credit score updates, hybrid outlier filtering, dynamic scaling factors, etc.) remain, but now integrate more tightly with the new ledger structure.

2. Modules and Submodules

2.1. Smart Contract Module (On-Chain Logic)

Purpose: Deploy Solidity contracts on the Ethereum TestNet to handle:

- **Wallet Linking & Verification:** Proof-of-ownership challenges and one-to-one mapping.
- **Fee & Reward Mechanism:** Verification fee collection and burn (unchanged).
- **Bridging Contracts:** Lock and mint tokens during cross-chain transfers between L1 and the COSINE L2 ledger.

Tools: Solidity, Truffle/Hardhat, Remix IDE.

Status (Updated):

- Added bridging contracts to facilitate cross-chain token transfers.
- Maintains essential on-chain logic for L1 operations, while the new COSINE L2 ledger manages most internal state and user data.

2.2. Validator Node Module

Purpose: Each validator node (packaged in a Docker container) is the core engine of the COSINE L2. Key responsibilities include:

- **Ledger Management (New)**: Maintaining the **WST** (Wallet State Trie), **GSO** (Global State Object), and **L1T** (Linked L1 Trie).
- **Transaction Processing (New)**: Validating and applying transactions (votes, token transfers, linking, bridging, etc.).
- **Block Production (New)**: Assembling transactions into blocks, computing updated state roots, and proposing blocks for consensus.
- **Consensus & VRF-Based Subset Selection**: Selecting a subset of validators to propose or validate blocks, also used for outlier filtering in credit score updates.
- **Credit Score Update Engine**: Processing votes, association risk penalties, and rehabilitation events to update wallet credit scores in the WST.
- **Hybrid Outlier Filtering**: Combining mean-based and median-based (MAD) measures, with dynamic thresholds (Kalman filter).
- **Dynamic Scaling Factor Feedback Loop**: Using Kalman filters to adjust penalty/reward factors, now stored in the GSO.

- **Reward Distribution & Cost Accumulation**: Tracking rewards and costs within the ledger.
- **Validator Reward & Staking Module**: Managing staking data in the WST/GSO (minimum 100,000 COSINE tokens required).
- **P2P Networking & Communication**: Exchanging blocks, transactions, and state among validators.

Language Options and Libraries (Validator Node):

Submodule	**Go Libraries**	**Rust Libraries**
Ledger Management	`go-ethereum/trie`, custom	`substrate-trie`, custom
Transaction Processing	Custom logic, `go-ethereum`	Custom logic, `substrate`
Block Production	Custom logic, `go-libp2p`	Custom logic, `libp2p`
Consensus & VRF (subset selection)	`go-libp2p`, `go-ethereum`	`libp2p`, Substrate
Credit Score Engine (with Hybrid Filter)	`gonum`, custom	`ndarray`, `nalgebra`
Kalman Filter-based Scaling	`gonum/mat` (custom KF)	`nalgebra` or `ndarray` (custom KF)
Reward & Cost Tracking	Custom logic, `gonum`	Custom logic, `ndarray`
Staking & Reward	`go-ethereum` integration	`ethers-rs` integration
P2P Networking	`go-libp2p`	`libp2p`

2.2.1. Ledger Management Submodule (New)

Function:

- Maintains the Wallet State Trie (WST), storing balance, credit score, and other wallet-specific data.
- Maintains the Global State Object (GSO), which holds protocol-wide parameters (e.g., dynamic scaling factors, total staked).
- Maintains the Linked L1 Trie (L1T) to enforce one-to-one mapping from L1 addresses to L2 wallets.
- Computes and verifies Merkle Patricia Trie (MPT) structures for the WST and L1T.
- Serializes and updates the GSO.

2.2.2. Transaction Processing Submodule (New)

Function:

- Receives and validates incoming transactions (e.g., token transfers, votes, linking requests, bridging operations).
- Applies transaction logic to update the WST, GSO, and/or L1T accordingly.
- Enforces rules for credit scoring (increment/decrement), bridging (lock/mint), staking, and voting.

2.2.3. Block Production Submodule (New)

Function:

- Collects valid transactions from the network.
- Applies them in a deterministic order, updating the state tries.
- Computes new state roots after transaction application.
- Generates a block with:
  - Previous block hash
  - State roots (WST root, L1T root, GSO root)
  - Timestamp, VRF output, block proposer signature
  - Transaction list
- Broadcasts the proposed block to other validators for consensus.

2.2.4. Consensus Submodule (Updated)

Function:

- Uses VRF-based subset selection to decide which validators propose and validate blocks.
- Incorporates the existing Hybrid Outlier Filtering for *credit score updates inside each block*, ensuring malicious outlier proposals are disregarded.
- Approves or rejects proposed blocks, finalizing updates to the ledger.

2.2.5. Credit Score Update Engine (Adapted)

Function:

- Processes sequential voting events, association penalties, and rehabilitation votes *within the ledger context.*
- Updates each wallet's credit score in the WST.

2.2.6. Hybrid Outlier Filtering Submodule (Unchanged Logic, New Integration)

Function:

- Computes mean/standard deviation and median/MAD for proposed credit score updates.
- Dynamically adjusts threshold parameters ($\tau(t)$, $k(t)$) using Kalman filters stored in the GSO.
- Filters out suspicious proposals during block validation.

2.2.7. Dynamic Scaling Factor Feedback Loop Submodule (Unchanged Logic, New Integration)

Function:

- Tracks observed vs. expected impacts of credit score shifts ($\Delta_{\text{vote}}, \Delta_{\text{assoc}}, \Delta_{\text{rehab}}$).
- Uses Kalman filters (parameters in the GSO) to refine $\{K_{\text{neg}}, K_{\text{pos}}, K_{\text{assoc}}, K_{\text{rehab}}\}$.
- Ensures penalty and reward factors remain adaptive to real-time network conditions.

2.2.8. Reward Distribution & Cost Accumulation (Unchanged Logic, New Integration)

Function:

- Rewards validators for proposing blocks and includes rewards for inlier proposals.
- Aggregates per-wallet costs, stored in the WST, for fee recoup.

2.3. Wallet Linking & Cross-Chain Module

Purpose: Enable users to link external L1 wallets (Ethereum and Bitcoin) to COSINE L2 wallets and manage bridging operations.

- **Ethereum Wallet Linking**: Uses signed challenges; integrates with bridging contracts on Ethereum.
- **Bitcoin Wallet Linking**: Verifies address ownership; enforces unique mapping in the L1T.
- **Association Risk Analysis**: Implements random time windows, random hop limits, and multi-hop risk calculations that feed into credit score logic.

Changes:

- Linking operations now update the WST and L1T within the validator-managed ledger (rather than storing data on IPFS).
- Bridging operations for token transfers involve locking tokens on L1 and minting them on L2 (and vice versa), reflected in the **Bridge Operation Submodule** described below.

Tools:

Submodule	**Go Libraries**	**Rust Libraries**
Ethereum Linking	`go-ethereum`	`ethers-rs`
Bitcoin Linking	`btcsuite/btcd`	`rust-bitcoin`
Association Analysis	Custom (with `gonum`)	`ndarray/nalgebra`

2.3.1. Bridge Operation Submodule (New)

Function:

- Coordinates with L1 bridging contracts to lock (or unlock) tokens on Ethereum or solana and for bitcoin no tokens can be deployed since it is a non programmable blockchain so only wallet linking can happen.
- Mints (or burns) corresponding L2 tokens on COSINE's ledger, updating the WST.
- Updates the **GSO** with global bridging statistics (e.g., total minted tokens, total burned tokens).

Status: Updated to include new bridging logic for transferring tokens between L1 and L2.

2.4. Voting, Governance & Credit Scoring Module

Purpose: Manage vote events, update voter reputation, process rehabilitation votes, and compute the unified credit score update:

$$C_W^{(t+1)} = C_W^{(t)} + \Delta_{\text{vote}} + \Delta_{\text{assoc}} + \Delta_{\text{rehab}}.$$

- **Vote Event Processor**: Records votes, updates the WST, and uses the *Hybrid Outlier Filter* to discard malicious proposals.
- **Reputation Management**: Adjusts voter reputation (in the WST) based on alignment with consensus.
- **Cosine Similarity Calculator**: Computes the cosine similarity between normalized vectors for verification.

Changes:

- All vote data and score updates are committed to the L2 ledger.
- Dynamic scaling factors ($K_{\text{neg}}, K_{\text{pos}}, K_{\text{assoc}}, K_{\text{rehab}}$) are retrieved from the GSO, updated via Kalman filters.

Library Suggestions:

Submodule	**Go Libraries**	**Rust Libraries**
Vote Processor	Custom with `gonum`	`ndarray/nalgebra`
Reputation Management	Custom logic	Custom logic
Cosine Similarity	`gonum` or custom	`nalgebra` or `ndarray`

2.5. Cross-Chain Interaction Module

Purpose: Facilitate borrowing, lending, and bridging across chains by linking/verify L1 wallets and L2 wallet states.

- **Bridge Contracts & Mapping**: On Ethereum, lock tokens and mint/burn them on L2.
- **Verification & Feedback Engine**: Implements cosine similarity verification and processes post-transaction feedback into the L2 ledger.

Changes:

- Fully integrated bridging logic now updates the ledger: *Bridge Operation Submodule* ensures consistency of token balances in the WST and bridging data in the GSO.
- Cross-chain operations rely on validated transactions in each L2 block.

Submodule	**Go Libraries**	**Rust Libraries**
Bridge Operation	`go-ethereum`, custom	`ethers-rs`, custom
Verification Engine	Custom with `gonum`	`ndarray`

2.6. Decentralized Data Storage Module (Removed / Repurposed)

Previous Purpose: Used IPFS to archive credit score data, validator state, and mapping files.

Changes:

- **L2 Ledger :** The primary storage mechanism is now the L2 ledger, maintained by validators.
- IPFS may be optionally used for off-chain archival or historical data, but is no longer part of the MVP's core architecture.

Status: *Removed or significantly reduced* in scope. All critical state is now stored in the ledger structures (WST, GSO, L1T, blocks).

2.7. API / CLI Interface Module

Purpose: Provide user and validator interactions via CLI and RESTful API.

- **CLI Application**: For wallet creation, linking, voting, token transfers, bridging operations, etc.
- **RESTful API Server**: Endpoints for querying ledger state (credit scores, token balances, bridging info) and submitting transactions.

Changes:

- Expanded to allow *transaction submission* directly to validator nodes, which process and integrate these transactions into blocks.
- Additional endpoints for bridging operations (e.g., lock, mint, burn, redeem).

Submodule	**Go Libraries**	**Rust Libraries**
CLI Application	`cobra`	`clap`
RESTful API Server	`gin`, `echo`	`actix-web`, `rocket`

2.8. Orchestration & Deployment Module

Purpose: Package and deploy validator nodes using Docker.

- **Docker Container Management**: Scripts and tools for building and running containers.
- **Virtual Network Setup**: Configures a P2P network among containers for block and transaction propagation.

Status: *Unchanged* in essence, but may require additional configuration for:

- Ledger parameters (e.g., block production intervals, trie settings).
- Bridging modules (e.g., environment variables for the bridging contract addresses).

Submodule	**Go Libraries**	**Rust Libraries**
Docker Integration	`docker/docker` SDK	Docker SDK bindings / CLI wrappers

2.9. Supporting Modules

Purpose: Provide logging, monitoring, and configuration management.

- **Logging & Monitoring**: Centralized logging, metrics collection (e.g., block production, consensus metrics, bridging stats).
- **Configuration Management**: Dynamic parameter management, now including ledger and bridging parameters (e.g., block intervals, bridging contract addresses).

Status: Updated.

- Additional monitoring required for the ledger components: WST, GSO, L1T, block production stats, etc.
- Configuration files extended to include ledger- and bridging-specific parameters.

Functionality	**Go Libraries**	**Rust Libraries**
Logging	`logrus`, `zap`	`log`, `env_logger`, `slog`
Configuration	`viper`	`config`
Monitoring	Prometheus client libraries	`prometheus` crate

3. Updated Deployment and Data Flow Summary

User Interaction:

- Users create L2 wallets (via CLI or API) and link their L1 wallets (Ethereum/Bitcoin) using proof-of-ownership.
- Users submit transactions (e.g., token transfers, votes, bridging operations) to validators.

Smart Contract Layer:

- Ethereum-based contracts handle bridging (locking/minting) and linking logic, plus fee and reward mechanics.
- The bulk of stateful logic now resides on the COSINE L2 ledger, maintained by validator nodes.

Validator Node Processing:

- **Ledger Management**: Each node maintains the WST (wallet data), GSO (global parameters), and L1T (L1-to-L2 mappings) in Merkle tries.
- **Transaction Processing**: Nodes validate incoming transactions and update the ledger state.
- **Block Production**: Selected validators propose blocks containing valid transactions; the new state roots are computed after applying each transaction.
- **Consensus**: VRF-based subset selection is used to finalize blocks. Hybrid outlier filtering is integrated for credit score proposals.
- **Credit Score Engine**: Applies votes, association penalties, and rehab events to the WST within each block.
- **Reward Distribution**: Allocates block rewards and manages validator staking in the WST/GSO.

Cross-Chain Interaction:

- Bridge operations lock or unlock tokens on L1 (via bridging contracts) and mint or burn corresponding tokens on the L2 ledger.
- Linked L1 addresses are enforced by the L1T in the ledger, ensuring one-to-one mapping to L2 wallets.

API/CLI Interface:

- Users submit transactions (votes, bridging requests, credit score queries, etc.) via RESTful endpoints or CLI commands.
- Validators expose endpoints for real-time ledger state queries (e.g., credit score, token balances, bridging status).

Orchestration:

- Validator nodes run in Docker containers on a P2P network, with each node maintaining a copy of the ledger.
- Configuration parameters (e.g., block interval) can be set per environment (testnet or dev).

4. Updated Visual Architecture Diagram

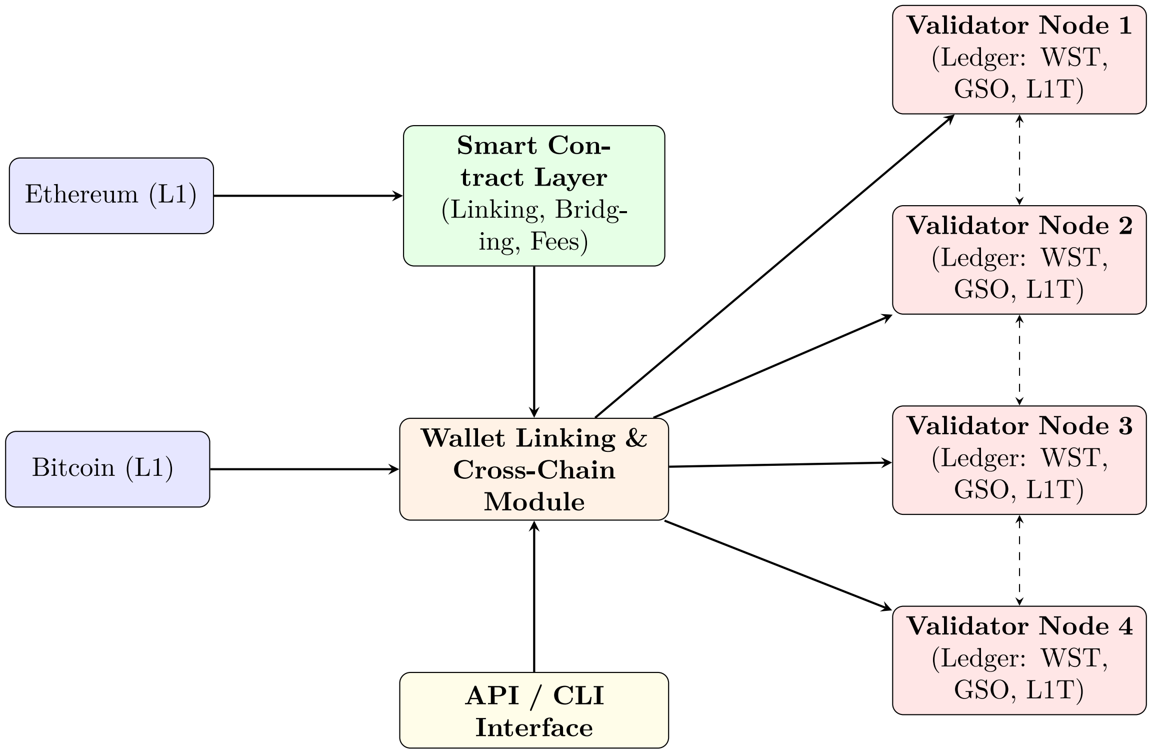
Validator Node 1 (Ledger: WST, GSO, L1T)
Smart Contract Layer (Linking, Bridging, Fees)
Ethereum (L1)
Validator Node 2 (Ledger: WST, GSO, L1T)
Bitcoin (L1)
Wallet Linking & Cross-Chain Module
Validator Node 3 (Ledger: WST, GSO, L1T)
Validator Node 4 (Ledger: WST, GSO, L1T)
API / CLI Interface

Figure 1: Updated COSINE MVP Architecture Diagram with L2 Ledger

5. Conclusion

The latest COSINE whitepaper's introduction of an L2 ledger, managed by validators, fundamentally alters the storage and update mechanisms of the system. IPFS is replaced (or relegated to an optional role) by a robust on-chain ledger approach, built around:

- A **Wallet State Trie (WST)** storing per-wallet data (credit scores, balances, etc.).
- A **Global State Object (GSO)** holding protocol-wide parameters, including dynamic scaling factors managed by Kalman filters.
- A **Linked L1 Trie (L1T)** enforcing strict one-to-one mappings between L1 addresses and COSINE L2 wallets.
- **Block Production and Transaction Processing** submodules, ensuring consensus-based updates to the state.

Cross-chain bridging is now integrated at both the L1 (via new bridging contracts) and L2 (via the Bridge Operation Submodule), allowing users to lock tokens on Ethereum and mint them on the COSINE L2 ledger.

All credit scoring, hybrid outlier filtering, reward distribution, and staking functionalities remain in place but now read and write directly to the ledger data structures rather than external storage. This alignment with the new "Decentralized Storage and Updates" architecture ensures a self-contained, tamper-proof, and scalable solution for managing credit scores, wallet states, and cross-chain assets, all while retaining the familiar MVP structure and user-facing features.